REMARKS

Status:

Claims 1-12 stand rejected under 35 U.S.C. §103(a) as being unpatentable over the teaching of U. S. Pat. No. 5,923,756 to Shambroom, considered with the teaching of Schneier Applied Cryptography in view of the teaching of Balenson, " Privacy Enhancement for Internet Electronic Mail: Algorithms, Modes, and Identifiers", Network Working Group, Request For Comments (RFC) 1423, February 1993.

Claims 1-12 as amended are presented for reconsideration as explained in the analysis that follows.

Analysis:

To start, it is helpful to look to the Schneier teaching at page 574 and the box representation of an X.509 certificate at Figure24.2. The "Algorithm Identifier" box shows an "Algorithm" entry and a "Parameters" entry. The "Subject's Public Key" box shows an "Algorithm" entry, a "Parameters" entry and a "Public Key" entry.

With this graphic in mind, more authoritative detail is provided regarding X.509 certificates at "RFC 2459 Internet X.509 Public Key Infrastructure (January 1999) which is available at http://www.ietf.org/rfc/ (Selected pages are included in Appendix A hereto).

At §4.1.2.7 "Subject Public Key Info" is discussed as follows: "*This field is used* ***to carry the public key and identify the algorithm with which the key is used.*** *The algorithm is identified using the Algorithm Identifier structure specified in section 4.1.1.2. The object identifiers for the supported algorithms and the methods for encoding the public key materials (public key and parameters) are specified in section 7.3.*"(italics added to indicate quotation, bolding added for emphasis)

There may be a host of alternatives for algorithms but it is indicated that for a conforming certificate, you get to select ONE with which the public key identified in the certificate is used.

Again, this is not to say there are no alternatives. Indeed, the Balenson teaching of RFC 1423 appears to be an effort to add more cryptographic algorithms. More to pick from. But, in an X.509 certificate, one algorithm is called for to be associated with the public key that is identified (see again the box representation of X.509 in the Schneier teaching).

As indicated in Applicants specification, there are times when the sender may wish to communicate (see Alice and Bob discussion at p. 575 of the Schneier teaching) but not know the algorithms the other party supports. A safe choice is a common one such as RSA, but that might be less secure than other alternatives.

Applicant has recognized that by using the certificate extensions described at §4.1.2.9 of RFC 2459 (see Appendix A) to provide lists of public keys, each with associated encryption algorithm, and a corresponding signature, the receiver may be enabled to select a more desirable trusted algorithm supported by the sender. Hence a certificate, by the use of permitted (under X509 v.3) extensions, but for an unintended purpose, may be able to offer plural public keys with associated cryptographic algorithms. Further, by making the certificate extensions non-critical (see section 4.2 of RFC 2459), receivers unable to process the extensions may operate with the certificate and ignore the extensions.

Now considering the applied references, it appears the Schneier teaching describes the usual certificates of the RFC 2459 document. The Balenson teaching appears related to improved algorithms and the ASN.1 identifiers mentioned are the same as called for in RFC 2459.

The Shambroom teaching indicates that it identifies plural algorithms (col. 10, lines 32-35) but doesn't teach how. It says the certificate "may resemble an ITU X.509 certificate" What does that mean? It indicates there is but one public key (Shambroom col. 10, line 32). Where is each

algorithm associated with a public key that is signed for security as called for in Applicant's claims? The Shambroom teaching seems a mere data list of algorithms supported. Not in extensions and not associated with public keys or signatures. And, the Schneier and Balanson teachings, as indicated above, do not overcome the deficiencies of Shambroom In accordance with this analysis, it is believed that the prior art taken singly or together does not teach or suggest applicants approach to expanding the public key information, including associated algorithms and signatures, communicated by an X.509 certificate through the use of certificate extensions. Certificate extensions, a capability authorized in the X.509 v.3 specification detailed in RFC 2459.

Applicant has amended the claims to more clearly emphasize the X.509 certificate structure and the inventive use of certificate extensions to allow an expansion beyond the limits of the base X.509 certificate. This expansion involving respective public keys associated with cryptographic algorithms and identifiable with respective signatures to support establishment of a trusted conversation.

In accordance with the foregoing, it is believed that the subject Application clearly identifies inventive subject matter not taught or suggested in the prior art. Hence Applicant respectfully solicits withdrawal of the rejection of claims under 35 U.S.C. §103(a) and early notice that this case has been placed in condition for allowance.

Respectfully Submitted,

_____

George E. Grosser
Reg. No. 25,629I

c/o IBM Corp.
Dept. T81/Bldg. 503 PO Box 12195
Research Triangle Park, NC 27709
(919)968-7847 Fax 919-254-4330
EMAIL: gegch@prodigy.net

Appendix A

RFC 2459 Internet Public Key Infrastructure
January 1999 §§4.0-4.2.1.5

4 Certificate and Certificate Extensions Profile

This section presents a profile for public key certificates that will foster interoperability and a reusable PKI. This section is based upon the X.509 v3 certificate format and the standard certificate extensions defined in [X.509]. The ISO/IEC/ITU documents use the 1993 version of ASN.1; while this document uses the 1988 ASN.1 syntax, the encoded certificate and standard extensions are equivalent. This section also defines private extensions required to support a PKI for the Internet community.

Certificates may be used in a wide range of applications and environments covering a broad spectrum of interoperability goals and a broader spectrum of operational and assurance requirements. The goal of this document is to establish a common baseline for generic applications requiring broad interoperability and limited special purpose requirements. In particular, the emphasis will be on supporting the use of X.509 v3 certificates for informal Internet electronic mail, IPsec, and WWW applications.

4.1 Basic Certificate Fields

The X.509 v3 certificate basic syntax is as follows. For signature calculation, the certificate is encoded using the ASN.1 distinguished encoding rules (DER) [X.208]. ASN.1 DER encoding is a tag, length, value encoding system for each element.

```
Certificate  ::=  SEQUENCE  {
     tbsCertificate       TBSCertificate,
     signatureAlgorithm   AlgorithmIdentifier,
     signatureValue       BIT STRING  }

TBSCertificate  ::=  SEQUENCE  {
     version         [0]  EXPLICIT Version DEFAULT v1,
     serialNumber         CertificateSerialNumber,
     signature            AlgorithmIdentifier,
     issuer               Name,
     validity             Validity,
     subject              Name,
     subjectPublicKeyInfo SubjectPublicKeyInfo,
     issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version shall be v2 or v3
     subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version shall be v2 or v3
     extensions      [3]  EXPLICIT Extensions OPTIONAL
                          -- If present, version shall be v3
     }
```

```
Version  ::=  INTEGER  {  v1(0), v2(1), v3(2)  }

CertificateSerialNumber  ::=  INTEGER

Validity ::= SEQUENCE {
     notBefore      Time,
     notAfter       Time }

Time ::= CHOICE {
     utcTime        UTCTime,
     generalTime    GeneralizedTime }

UniqueIdentifier  ::=  BIT STRING

SubjectPublicKeyInfo  ::=  SEQUENCE  {
     algorithm            AlgorithmIdentifier,
     subjectPublicKey     BIT STRING  }

Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension

Extension  ::=  SEQUENCE  {
     extnID       OBJECT IDENTIFIER,
     critical     BOOLEAN DEFAULT FALSE,
     extnValue    OCTET STRING  }
```

The following items describe the X.509 v3 certificate for use in the Internet.

4.1.1 Certificate Fields

The Certificate is a SEQUENCE of three required fields. The fields are described in detail in the following subsections.

4.1.1.1 tbsCertificate

The field contains the names of the subject and issuer, a public key associated with the subject, a validity period, and other associated information. The fields are described in detail in section 4.1.2; the tbscertificate may also include extensions which are described in section 4.2.

4.1.1.2 signatureAlgorithm

The signatureAlgorithm field contains the identifier for the cryptographic algorithm used by the CA to sign this certificate. Section 7.2 lists the supported signature algorithms.

An algorithm identifier is defined by the following ASN.1 structure:

```
AlgorithmIdentifier  ::=  SEQUENCE  {
     algorithm               OBJECT IDENTIFIER,
     parameters              ANY DEFINED BY algorithm OPTIONAL  }
```

The algorithm identifier is used to identify a cryptographic algorithm. The OBJECT IDENTIFIER component identifies the algorithm (such as DSA with SHA-1). The contents of the optional parameters field will vary according to the algorithm identified. Section 7.2 lists the supported algorithms for this specification.

This field MUST contain the same algorithm identifier as the signature field in the sequence tbsCertificate (see sec. 4.1.2.3).

4.1.1.3 signatureValue

The signatureValue field contains a digital signature computed upon the ASN.1 DER encoded tbsCertificate. The ASN.1 DER encoded tbsCertificate is used as the input to the signature function. This signature value is then ASN.1 encoded as a BIT STRING and included in the Certificate's signature field. The details of this process are specified for each of the supported algorithms in Section 7.2.

By generating this signature, a CA certifies the validity of the information in the tbsCertificate field. In particular, the CA certifies the binding between the public key material and the subject of the certificate.

4.1.2 TBSCertificate

The sequence TBSCertificate contains information associated with the subject of the certificate and the CA who issued it. Every TBSCertificate contains the names of the subject and issuer, a public key associated with the subject, a validity period, a version number, and a serial number; some may contain optional unique identifier fields. The remainder of this section describes the syntax and semantics of these fields. A TBSCertificate may also include extensions. Extensions for the Internet PKI are described in Section 4.2.

4.1.2.1 Version

This field describes the version of the encoded certificate. When extensions are used, as expected in this profile, use X.509 version 3 (value is 2). If no extensions are present, but a UniqueIdentifier is present, use version 2 (value is 1). If only basic fields are present, use version 1 (the value is omitted from the certificate as the default value).

Implementations SHOULD be prepared to accept any version certificate. At a minimum, conforming implementations MUST recognize version 3 certificates.

Generation of version 2 certificates is not expected by implementations based on this profile.

4.1.2.2 Serial number

The serial number is an integer assigned by the CA to each certificate. It MUST be unique for each certificate issued by a given CA (i.e., the issuer name and serial number identify a unique certificate).

4.1.2.3 Signature

This field contains the algorithm identifier for the algorithm used by the CA to sign the certificate.

This field MUST contain the same algorithm identifier as the signatureAlgorithm field in the sequence Certificate (see sec. 4.1.1.2). The contents of the optional parameters field will vary according to the algorithm identified. Section 7.2 lists the supported signature algorithms.

4.1.2.4 Issuer

The issuer field identifies the entity who has signed and issued the certificate. The issuer field MUST contain a non-empty distinguished name (DN). The issuer field is defined as the X.501 type Name. [X.501] Name is defined by the following ASN.1 structures:

```
Name ::= CHOICE {
  RDNSequence }

RDNSequence ::= SEQUENCE OF RelativeDistinguishedName

RelativeDistinguishedName ::=
  SET OF AttributeTypeAndValue

AttributeTypeAndValue ::= SEQUENCE {
  type     AttributeType,
  value    AttributeValue }

AttributeType ::= OBJECT IDENTIFIER

AttributeValue ::= ANY DEFINED BY AttributeType
```



```
DirectoryString ::= CHOICE {
      teletexString           TeletexString (SIZE (1..MAX)),
      printableString         PrintableString (SIZE (1..MAX)),
      universalString         UniversalString (SIZE (1..MAX)),
      utf8String              UTF8String (SIZE (1.. MAX)),
      bmpString               BMPString (SIZE (1..MAX)) }
```

The Name describes a hierarchical name composed of attributes, such

as country name, and corresponding values, such as US. The type of the component AttributeValue is determined by the AttributeType; in general it will be a DirectoryString.

The DirectoryString type is defined as a choice of PrintableString, TeletexString, BMPString, UTF8String, and UniversalString. The UTF8String encoding is the preferred encoding, and all certificates issued after December 31, 2003 MUST use the UTF8String encoding of DirectoryString (except as noted below). Until that date, conforming CAs MUST choose from the following options when creating a distinguished name, including their own:

(a) if the character set is sufficient, the string MAY be represented as a PrintableString;

(b) failing (a), if the BMPString character set is sufficient the string MAY be represented as a BMPString; and

(c) failing (a) and (b), the string MUST be represented as a UTF8String. If (a) or (b) is satisfied, the CA MAY still choose to represent the string as a UTF8String.

Exceptions to the December 31, 2003 UTF8 encoding requirements are as follows:

(a) CAs MAY issue "name rollover" certificates to support an orderly migration to UTF8String encoding. Such certificates would include the CA's UTF8String encoded name as issuer and and the old name encoding as subject, or vice-versa.

(b) As stated in section 4.1.2.6, the subject field MUST be populated with a non-empty distinguished name matching the contents of the issuer field in all certificates issued by the subject CA regardless of encoding.

The TeletexString and UniversalString are included for backward compatibility, and should not be used for certificates for new subjects. However, these types may be used in certificates where the name was previously established. Certificate users SHOULD be prepared to receive certificates with these types.

In addition, many legacy implementations support names encoded in the ISO 8859-1 character set (Latin1String) but tag them as TeletexString. The Latin1String includes characters used in Western European countries which are not part of the TeletexString charcter set. Implementations that process TeletexString SHOULD be prepared to handle the entire ISO 8859-1 character set.[ISO 8859-1]

As noted above, distinguished names are composed of attributes. This specification does not restrict the set of attribute types that may appear in names. However, conforming implementations MUST be prepared to receive certificates with issuer names containing the set of attribute types defined below. This specification also recommends

support for additional attribute types.

Standard sets of attributes have been defined in the X.500 series of specifications. [X.520] Implementations of this specification MUST be prepared to receive the following standard attribute types in issuer names: country, organization, organizational-unit, distinguished name qualifier, state or province name, and common name (e.g., "Susan Housley"). In addition, implementations of this specification SHOULD be prepared to receive the following standard attribute types in issuer names: locality, title, surname, given name, initials, and generation qualifier (e.g., "Jr.", "3rd", or "IV"). The syntax and associated object identifiers (OIDs) for these attribute types are provided in the ASN.1 modules in Appendices A and B.

In addition, implementations of this specification MUST be prepared to receive the domainComponent attribute, as defined in [RFC 2247]. The Domain (Nameserver) System (DNS) provides a hierarchical resource labeling system. This attribute provides is a convenient mechanism for organizations that wish to use DNs that parallel their DNS names. This is not a replacement for the dNSName component of the alternative name field. Implementations are not required to convert such names into DNS names. The syntax and associated OID for this attribute type is provided in the ASN.1 modules in Appendices A and B.

Certificate users MUST be prepared to process the issuer distinguished name and subject distinguished name (see sec. 4.1.2.6) fields to perform name chaining for certification path validation (see section 6). Name chaining is performed by matching the issuer distinguished name in one certificate with the subject name in a CA certificate.

This specification requires only a subset of the name comparison functionality specified in the X.500 series of specifications. The requirements for conforming implementations are as follows:

(a) attribute values encoded in different types (e.g., PrintableString and BMPString) may be assumed to represent different strings;

(b) attribute values in types other than PrintableString are case sensitive (this permits matching of attribute values as binary objects);

(c) attribute values in PrintableString are not case sensitive (e.g., "Marianne Swanson" is the same as "MARIANNE SWANSON"); and

(d) attribute values in PrintableString are compared after removing leading and trailing white space and converting internal substrings of one or more consecutive white space characters to a single space.

These name comparison rules permit a certificate user to validate certificates issued using languages or encodings unfamiliar to the certificate user.

In addition, implementations of this specification MAY use these comparison rules to process unfamiliar attribute types for name chaining. This allows implementations to process certificates with unfamiliar attributes in the issuer name.

Note that the comparison rules defined in the X.500 series of specifications indicate that the character sets used to encode data in distinguished names are irrelevant. The characters themselves are compared without regard to encoding. Implementations of the profile are permitted to use the comparison algorithm defined in the X.500 series. Such an implementation will recognize a superset of name matches recognized by the algorithm specified above.

4.1.2.5 Validity

The certificate validity period is the time interval during which the CA warrants that it will maintain information about the status of the certificate. The field is represented as a SEQUENCE of two dates: the date on which the certificate validity period begins (notBefore) and the date on which the certificate validity period ends (notAfter). Both notBefore and notAfter may be encoded as UTCTime or GeneralizedTime.

CAs conforming to this profile MUST always encode certificate validity dates through the year 2049 as UTCTime; certificate validity dates in 2050 or later MUST be encoded as GeneralizedTime.

4.1.2.5.1 UTCTime

The universal time type, UTCTime, is a standard ASN.1 type intended for international applications where local time alone is not adequate. UTCTime specifies the year through the two low order digits and time is specified to the precision of one minute or one second. UTCTime includes either Z (for Zulu, or Greenwich Mean Time) or a time differential.

For the purposes of this profile, UTCTime values MUST be expressed Greenwich Mean Time (Zulu) and MUST include seconds (i.e., times are YYMMDDHHMMSSZ), even where the number of seconds is zero. Conforming systems MUST interpret the year field (YY) as follows:

Where YY is greater than or equal to 50, the year shall be interpreted as 19YY; and

Where YY is less than 50, the year shall be interpreted as 20YY.

4.1.2.5.2 GeneralizedTime

The generalized time type, GeneralizedTime, is a standard ASN.1 type for variable precision representation of time. Optionally, the GeneralizedTime field can include a representation of the time differential between local and Greenwich Mean Time.

For the purposes of this profile, GeneralizedTime values MUST be expressed Greenwich Mean Time (Zulu) and MUST include seconds (i.e., times are YYYYMMDDHHMMSSZ), even where the number of seconds is zero. GeneralizedTime values MUST NOT include fractional seconds.

4.1.2.6 Subject

The subject field identifies the entity associated with the public key stored in the subject public key field. The subject name may be carried in the subject field and/or the subjectAltName extension. If the subject is a CA (e.g., the basic constraints extension, as discussed in 4.2.1.10, is present and the value of cA is TRUE,) then the subject field MUST be populated with a non-empty distinguished name matching the contents of the issuer field (see sec. 4.1.2.4) in all certificates issued by the subject CA. If subject naming information is present only in the subjectAltName extension (e.g., a key bound only to an email address or URI), then the subject name MUST be an empty sequence and the subjectAltName extension MUST be critical.

Where it is non-empty, the subject field MUST contain an X.500 distinguished name (DN). The DN MUST be unique for each subject entity certified by the one CA as defined by the issuer name field. A CA may issue more than one certificate with the same DN to the same subject entity.

The subject name field is defined as the X.501 type Name. Implementation requirements for this field are those defined for the issuer field (see sec. 4.1.2.4). When encoding attribute values of type DirectoryString, the encoding rules for the issuer field MUST be implemented. Implementations of this specification MUST be prepared to receive subject names containing the attribute types required for the issuer field. Implementations of this specification SHOULD be prepared to receive subject names containing the recommended attribute types for the issuer field. The syntax and associated object identifiers (OIDs) for these attribute types are provided in the ASN.1 modules in Appendices A and B. Implementations of this specification MAY use these comparison rules to process unfamiliar attribute types (i.e., for name chaining). This allows implementations to process certificates with unfamiliar attributes in the subject name.

In addition, legacy implementations exist where an RFC 822 name is embedded in the subject distinguished name as an EmailAddress

attribute. The attribute value for EmailAddress is of type IA5String to permit inclusion of the character '@', which is not part of the PrintableString character set. EmailAddress attribute values are not case sensitive (e.g., "fanfeedback@redsox.com" is the same as "FANFEEDBACK@REDSOX.COM").

Conforming implementations generating new certificates with electronic mail addresses MUST use the rfc822Name in the subject alternative name field (see sec. 4.2.1.7) to describe such identities. Simultaneous inclusion of the EmailAddress attribute in the subject distinguished name to support legacy implementations is deprecated but permitted.

4.1.2.7 Subject Public Key Info

This field is used to carry the public key and identify the algorithm with which the key is used. The algorithm is identified using the AlgorithmIdentifier structure specified in section 4.1.1.2. The object identifiers for the supported algorithms and the methods for encoding the public key materials (public key and parameters) are specified in section 7.3.

4.1.2.8 Unique Identifiers

These fields may only appear if the version is 2 or 3 (see sec. 4.1.2.1). The subject and issuer unique identifiers are present in the certificate to handle the possibility of reuse of subject and/or issuer names over time. This profile recommends that names not be reused for different entities and that Internet certificates not make use of unique identifiers. CAs conforming to this profile SHOULD NOT generate certificates with unique identifiers. Applications conforming to this profile SHOULD be capable of parsing unique identifiers and making comparisons.

4.1.2.9 Extensions

This field may only appear if the version is 3 (see sec. 4.1.2.1). If present, this field is a SEQUENCE of one or more certificate extensions. The format and content of certificate extensions in the Internet PKI is defined in section 4.2.

4.2 Standard Certificate Extensions

The extensions defined for X.509 v3 certificates provide methods for associating additional attributes with users or public keys and for managing the certification hierarchy. The X.509 v3 certificate format also allows communities to define private extensions to carry information unique to those communities. Each extension in a certificate may be designated as critical or non-critical. A certificate using system MUST reject the certificate if it encounters

a critical extension it does not recognize; however, a non-critical extension may be ignored if it is not recognized. The following sections present recommended extensions used within Internet certificates and standard locations for information. Communities may elect to use additional extensions; however, caution should be exercised in adopting any critical extensions in certificates which might prevent use in a general context.

Each extension includes an OID and an ASN.1 structure. When an extension appears in a certificate, the OID appears as the field extnID and the corresponding ASN.1 encoded structure is the value of the octet string extnValue. Only one instance of a particular extension may appear in a particular certificate. For example, a certificate may contain only one authority key identifier extension (see sec. 4.2.1.1). An extension includes the boolean critical, with a default value of FALSE. The text for each extension specifies the acceptable values for the critical field.

Conforming CAs MUST support key identifiers (see sec. 4.2.1.1 and 4.2.1.2), basic constraints (see sec. 4.2.1.10), key usage (see sec. 4.2.1.3), and certificate policies (see sec. 4.2.1.5) extensions. If the CA issues certificates with an empty sequence for the subject field, the CA MUST support the subject alternative name extension (see sec. 4.2.1.7). Support for the remaining extensions is OPTIONAL. Conforming CAs may support extensions that are not identified within this specification; certificate issuers are cautioned that marking such extensions as critical may inhibit interoperability.

At a minimum, applications conforming to this profile MUST recognize the extensions which must or may be critical in this specification. These extensions are: key usage (see sec. 4.2.1.3), certificate policies (see sec. 4.2.1.5), the subject alternative name (see sec. 4.2.1.7), basic constraints (see sec. 4.2.1.10), name constraints (see sec. 4.2.1.11), policy constraints (see sec. 4.2.1.12), and extended key usage (see sec. 4.2.1.13).

In addition, this profile RECOMMENDS application support for the authority and subject key identifier (see sec. 4.2.1.1 and 4.2.1.2) extensions.

4.2.1 Standard Extensions

This section identifies standard certificate extensions defined in [X.509] for use in the Internet PKI. Each extension is associated with an OID defined in [X.509]. These OIDs are members of the id-ce arc, which is defined by the following:

```
id-ce   OBJECT IDENTIFIER ::=  {joint iso ccitt(2) ds(5) 29}
```

4.2.1.1 Authority Key Identifier

The authority key identifier extension provides a means of identifying the public key corresponding to the private key used to sign a certificate. This extension is used where an issuer has multiple signing keys (either due to multiple concurrent key pairs or due to changeover). The identification may be based on either the key identifier (the subject key identifier in the issuer's certificate) or on the issuer name and serial number.

The keyIdentifier field of the authorityKeyIdentifier extension MUST be included in all certificates generated by conforming CAs to facilitate chain building. There is one exception; where a CA distributes its public key in the form of a "self-signed" certificate, the authority key identifier may be omitted. In this case, the subject and authority key identifiers would be identical.

The value of the keyIdentifier field SHOULD be derived from the public key used to verify the certificate's signature or a method that generates unique values. Two common methods for generating key identifiers from the public key are described in (sec. 4.2.1.2). One common method for generating unique values isdescribed in (sec. 4.2.1.2). Where a key identifier has not been previously established, this specification recommends use of one of these methods for generating keyIdentifiers.

This profile recommends support for the key identifier method by all certificate users.

This extension MUST NOT be marked critical.

```
id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::=  { id-ce 35 }

AuthorityKeyIdentifier ::= SEQUENCE {
   keyIdentifier             [0] KeyIdentifier           OPTIONAL,
   authorityCertIssuer       [1] GeneralNames            OPTIONAL,
   authorityCertSerialNumber [2] CertificateSerialNumber OPTIONAL  }

KeyIdentifier ::= OCTET STRING
```

4.2.1.2 Subject Key Identifier

The subject key identifier extension provides a means of identifying certificates that contain a particular public key.

To facilitate chain building, this extension MUST appear in all conforming CA certificates, that is, all certificates including the basic constraints extension (see sec. 4.2.1.10) where the value of cA is TRUE. The value of the subject key identifier MUST be the value placed in the key identifier field of the Authority Key Identifier extension (see sec. 4.2.1.1) of certificates issued by the subject of this certificate.

For CA certificates, subject key identifiers SHOULD be derived from the public key or a method that generates unique values. Two common methods for generating key identifiers from the public key are:

(1) The keyIdentifier is composed of the 160-bit SHA-1 hash of the value of the BIT STRING subjectPublicKey (excluding the tag, length, and number of unused bits).

(2) The keyIdentifier is composed of a four bit type field with the value 0100 followed by the least significant 60 bits of the SHA-1 hash of the value of the BIT STRING subjectPublicKey.

One common method for generating unique values is a monotomically increasing sequence of integers.

For end entity certificates, the subject key identifier extension provides a means for identifying certificates containing the particular public key used in an application. Where an end entity has obtained multiple certificates, especially from multiple CAs, the subject key identifier provides a means to quickly identify the set of certificates containing a particular public key. To assist applications in identificiation the appropriate end entity certificate, this extension SHOULD be included in all end entity certificates.

For end entity certificates, subject key identifiers SHOULD be derived from the public key. Two common methods for generating key identifiers from the public key are identifed above.

Where a key identifier has not been previously established, this specification recommends use of one of these methods for generating keyIdentifiers.

This extension MUST NOT be marked critical.

```
id-ce-subjectKeyIdentifier OBJECT IDENTIFIER ::=  { id-ce 14 }

SubjectKeyIdentifier ::= KeyIdentifier
```

4.2.1.3 Key Usage

The key usage extension defines the purpose (e.g., encipherment, signature, certificate signing) of the key contained in the certificate. The usage restriction might be employed when a key that could be used for more than one operation is to be restricted. For example, when an RSA key should be used only for signing, the digitalSignature and/or nonRepudiation bits would be asserted. Likewise, when an RSA key should be used only for key management, the keyEncipherment bit would be asserted. When used, this extension SHOULD be marked critical.

```
id-ce-keyUsage OBJECT IDENTIFIER ::=  { id-ce 15 }
```

```
KeyUsage ::= BIT STRING {
     digitalSignature        (0),
     nonRepudiation          (1),
     keyEncipherment         (2),
     dataEncipherment        (3),
     keyAgreement            (4),
     keyCertSign             (5),
```



```
     cRLSign                 (6),
     encipherOnly            (7),
     decipherOnly            (8) }
```

Bits in the KeyUsage type are used as follows:

The digitalSignature bit is asserted when the subject public key is used with a digital signature mechanism to support security services other than non-repudiation (bit 1), certificate signing (bit 5), or revocation information signing (bit 6). Digital signature mechanisms are often used for entity authentication and data origin authentication with integrity.

The nonRepudiation bit is asserted when the subject public key is used to verify digital signatures used to provide a non-repudiation service which protects against the signing entity falsely denying some action, excluding certificate or CRL signing.

The keyEncipherment bit is asserted when the subject public key is used for key transport. For example, when an RSA key is to be used for key management, then this bit shall asserted.

The dataEncipherment bit is asserted when the subject public key is used for enciphering user data, other than cryptographic keys.

The keyAgreement bit is asserted when the subject public key is used for key agreement. For example, when a Diffie-Hellman key is to be used for key management, then this bit shall asserted.

The keyCertSign bit is asserted when the subject public key is used for verifying a signature on certificates. This bit may only be asserted in CA certificates.

The cRLSign bit is asserted when the subject public key is used for verifying a signature on revocation information (e.g., a CRL).

The meaning of the encipherOnly bit is undefined in the absence of the keyAgreement bit. When the encipherOnly bit is asserted and the keyAgreement bit is also set, the subject public key may be used only for enciphering data while performing key agreement.

The meaning of the decipherOnly bit is undefined in the absence of the keyAgreement bit. When the decipherOnly bit is asserted and

the keyAgreement bit is also set, the subject public key may be used only for deciphering data while performing key agreement.

This profile does not restrict the combinations of bits that may be set in an instantiation of the keyUsage extension. However, appropriate values for keyUsage extensions for particular algorithms are specified in section 7.3.

4.2.1.4 Private Key Usage Period

This profile recommends against the use of this extension. CAs conforming to this profile MUST NOT generate certificates with critical private key usage period extensions.

The private key usage period extension allows the certificate issuer to specify a different validity period for the private key than the certificate. This extension is intended for use with digital signature keys. This extension consists of two optional components, notBefore and notAfter. The private key associated with the certificate should not be used to sign objects before or after the times specified by the two components, respectively. CAs conforming to this profile MUST NOT generate certificates with private key usage period extensions unless at least one of the two components is present.

Where used, notBefore and notAfter are represented as GeneralizedTime and MUST be specified and interpreted as defined in section 4.1.2.5.2.

```
id-ce-privateKeyUsagePeriod OBJECT IDENTIFIER ::=  { id-ce 16 }

PrivateKeyUsagePeriod ::= SEQUENCE {
     notBefore       [0]     GeneralizedTime OPTIONAL,
     notAfter        [1]     GeneralizedTime OPTIONAL }
```

4.2.1.5 Certificate Policies

The certificate policies extension contains a sequence of one or more policy information terms, each of which consists of an object identifier (OID) and optional qualifiers. These policy information terms indicate the policy under which the certificate has been issued and the purposes for which the certificate may be used. Optional qualifiers, which may be present, are not expected to change the definition of the policy.

Applications with specific policy requirements are expected to have a list of those policies which they will accept and to compare the policy OIDs in the certificate to that list. If this extension is critical, the path validation software MUST be able to interpret this extension (including the optional qualifier), or MUST reject the certificate.